**Association for Information Systems**
**AIS Electronic Library (AISeL)**

ICIS 1999 Proceedings

International Conference on Information Systems (ICIS)

December 1999

# An Empirical Study of Non-binary Genetic Algorithm-based Neural Approaches for Classification

Parag Pendharkar
*Penn State University at Harrisburg*

James Rodger
*Indiana University of Pennsylvania*

# AN EMPIRICAL STUDY OF NON-BINARY GENETIC ALGORITHM-BASED NEURAL APPROACHES FOR CLASSIFICATION

**Parag C. Pendharkar**
School of Business Administration
Penn State University at Harrisburg
U.S.A.

**James A. Rodger**
Eberly College of Business Administration
Indiana University of Pennsylvania
U.S.A.

## Abstract

In this paper, we describe a genetic algorithm (GA) based approach for learning connection weights for an artificial neural network (ANN). We use simulated data sets to compare the GA based approach for learning connection weights against the traditional back-propagation algorithm. Our results indicate that GA based training of ANN has a higher reliability (in terms of over-fitting the training data set) and predictive power than the traditional back-propagation algorithm.

**Keywords:** Neural networks, discriminant analysis, genetic algorithms

## 1. INTRODUCTION

The classification problem of assigning several observations into different disjoint groups plays an important role in business decision making. The binary classification problem, a sub-set of the classification problem, is a problem where the data are restricted to one of two disjoint groups. The binary classification problem (also called a two-group discriminant analysis problem) has wide applicability in problems ranging from credit scoring, default prediction and direct marketing to applications in finance and medical domains.

Several approaches were proposed for solving the binary classification problem. The approaches can be categorized as linear and non-linear discriminant analysis approaches. The linear approaches use a line or a plane to separate the two groups. Among the popular approaches for linear classification models are statistical discriminant analysis models (Fisher's discriminant analysis, LOGIT, PROBIT) and non-parametric discriminant analysis models such as genetic algorithm/artificial neural network based linear discriminant models. Non-linear approaches used for discriminant analysis fall into two categories: the connectionist approaches employing some form of artificial neural network (ANN) learning algorithm and the inductive learning models where the discriminant function is expressed in symbolic form using rules, decision trees, etc. The back-propagation ANN is the most commonly used connectionist scheme for non-linear discriminant analysis. Various induction algorithms have been suggested for classification, popular among them are CART, ID3, and CN2.

Among the popular supervised learning approaches for non-linear binary classification problems are ANN, genetic programming and ID3/C4.5. All of these approaches, however, have been criticized for over-fitting the training data set (Bhattacharyya and Pendharkar 1998). Recently, Pendharkar (1999) showed that adding random noise during the ANN training may alleviate the problem of over-fitting the training data set at the expense of a lower learning performance on the training data set. A few researchers criticized the training algorithm used in training ANN and proposed several modifications and alternate algorithms (Curry and Morgan 1997; Hung and Denton 1993; Sexton et al. 1998; Sexton, Dorsey and Johnson 1999). The popular alternate algorithm, GRG2, didn't outperform the ANNs in terms of learning (on training cases) and prediction accuracy (on unseen cases) (Hung and Denton 1993). GRG2 did outperform the ANN in convergence time, however.

In the current paper, we use the principles of evolution to train an ANN. Specifically, we use non-binary genetic algorithms (GAs) to learn the connection weights in an ANN. GAs are general-purpose evolutionary algorithms that can be used for optimization (Goldberg 1989). When compared to traditional optimization methods, GA provides heuristic optimal solutions. Although heuristic optimal solutions are less attractive when traditional optimization approaches are likely to find better solutions, they may be attractive when finding an optimal solution has a chance of over-fitting the training data set (Moriarty and Miikkulainen 1998). Further, the global and parallel nature of genetic search makes finding heuristic optimal solutions efficient when compared to the traditional local search based hill climbing and gradient descent optimization approaches such as back-propagation (Sexton, Dorsey and Johnson 1999). For complex search spaces, a problem that is easy for GA may be extremely difficult for steepest ascent optimization approaches (Wilson 1991). GA based learning of connection weights for an ANN has received some attention in the computer science and operations research literature (Miller, Todd and Hedge 1989; Moriarty and Miikkulainen 1998; Rooij, Jain and Johnson 1998; Sexton, Dorsey and Johnson 1999; Whitley and Hanson 1989). Most studies, however, used forecasting and function learning domain as an application and many studies used binary GAs. Binary GA representation has been criticized for longer convergence times and lack of solution accuracy (Janikow and Michalewicz 1991; Rooij, Jain and Johnson 1998). In the current research, we use GA based learning of connection weights for an ANN for a binary classification problem. Recent studies have shown that GA based learning of connection weights is a promising approach when compared to gradient descent approaches such as back-propagation and GRG2 and other heuristic approaches such as simulated annealing and tabu search (Sexton et al. 1998; Sexton, Dorsey and Johnson 1999). Unlike some of the previous studies, we use non-binary GA representation. We specifically investigate the impact of different types of design parameters (crossover operators), group distribution characteristics and group dispersion on learning and predictive performance of GA based ANN. The following are contributions of our research.

1. We study the learning and predictive performance (with special interest in over-fitting) of GA based ANN on training and unseen test cases under different data characteristics. Most other studies have focused on the training/learning performance of GA based ANN for a forecasting and function approximation problems. In this study, we investigate the predictive performance of GA based ANN for classification problem that has received little attention in the literature. Unlike the previous studies, which used root-mean-square (RMS) as the performance metric, we use number of correctly classified cases as our performance metric.

2. We investigate the performance of different crossover operators on the learning and predictive performance of GA based ANN for classification. Most studies in the past used only one type of crossover and benchmarked the performance of GA based ANN with other approaches.

3. Unlike studies in the past that were limited with few data sets and functions, we conduct extensive experiments to increase external validity of our study (limited by the characteristics of our data sets).

We use non-binary representation since there is evidence in the literature that non-binary GAs are intuitively appealing, and more efficient (in terms of use of computer memory and convergence times) (Rooij, Jain and Johnson 1998). The results of our study, therefore, should be interpreted in the realm of non-binary GAs.

The rest of the paper is organized as follows: Section 2 consists of introductory concepts and describes our GA implementation. Section 3 consists of a brief literature review and proposed hypotheses. Section 4 describes the data set used for our experiments. Section 5 details the results of our experiments. Section 6 concludes this paper with a summary and directions for future research.

## 2. INTRODUCTORY CONCEPTS AND DESCRIPTION OF OUR GA IMPLEMENTATION

Artificial neural networks and genetic algorithms were invented to mimic some of the phenomenon observed in Biology. The biological metaphor for ANNs is the human brain and the biological metaphor for GAs is evolution of a species. An ANN consists of different sets of neurons or nodes and the connections between one set of neurons to the other. Each connection between two nodes in different sets is assigned a weight that shows the strength of the connection. A connection with a positive weight is called an excitatory connection and a connection with a negative weight is called an inhibitory connection. The network of neurons and their connections is called the architecture of the ANN. Let A = $\{N_1, N_2, N_3\}$, B = $\{ N_4, N_5, N_6\}$, and C= $\{ N_7\}$ be three sets of nodes for an ANN. Set A is called the set of input nodes, set B is called the hidden set of nodes, and set C is called the set of output nodes. The cardinality of set A is equal to the number of input variables, the cardinality of set C is equal to number of output variables. Each connection can be view as a mapping from either an input node to a hidden node or from a hidden node to an output node. The general architecture of three sets of nodes is called a three-layer (of nodes) ANN. Figure 1 illustrates a three-layer ANN. Notice that the property $A \cap B \cap C = \Phi$ always hold true and the network is called feed-forward network. The connections in a feed-forward network are in one direction and the connections from A to B and from B to C are forward. The $w_{ij}$ are the weights that denote the strength of connection from node i to node j.

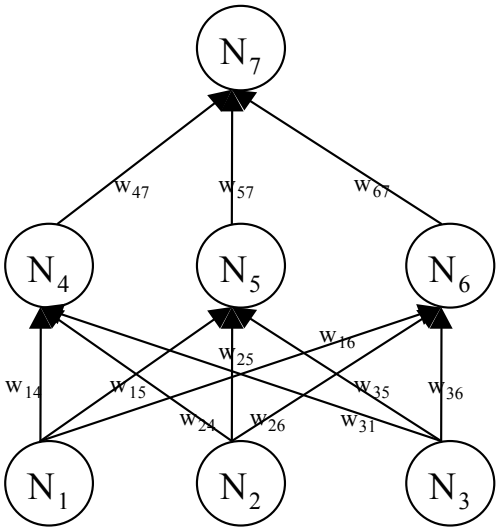


**Figure 1. A Three Layer (of Nodes) Artificial Neural Network**

Information is processed at each node in an ANN. For example, at hidden node $N_4$, the incoming signal vector (input) from the three nodes in the input set is multiplied by the strength of each connection and is added up. The result is passed through an activation function and the outcome is the activation for the node. If x represents the sum of the product of incoming signal vector and the strength of connection, then the activation, using logistic sigmoid activation function, can be represented by

$$f(x) = \frac{1}{1 + e^{-x}}$$

In the back-propagation algorithm based learning, the strengths of connections are randomly chosen. Based on the initial set of randomly chosen weights, the algorithm tries to minimize the following root-mean-square error (RMS):

$$E = \frac{1}{2}\sum_{n=0}^{n=N} \left\| t_n - o_n \right\|^2$$

where N is number of patterns in the training set, $t_n$ is the target output of the $n^{th}$ pattern, and $o_n$ is the actual output for the $n^{th}$ pattern. In each subsequent training step, the initial set of random connection weights (strength of connections) is adjusted toward the direction of maximum decrease of E, which is scaled by a learning rate lamda. Mathematically, an old weight $w_{old}$ is updated to its new value $w_{new}$ using the following equation:

$$w_{new} = w_{old} - \lambda \nabla E$$

$$\text{where } \nabla E = \left( \frac{\partial E}{\partial w_1}, \frac{\partial E}{\partial w_2}, \ldots, \frac{\partial E}{\partial w_n} \right)$$

One useful property of the sigmoid function is that

$$\frac{df(x)}{dx} = f(x)\big(1 - f(x)\big)$$

This means that the derivative (gradient) of the sigmoid function can be calculated by applying a simple multiplication and subtraction operator on the function itself. This property simplifies the computation of new weights from initial random values.

Genetic algorithms (GAs) use a *survival of the fittest strategy* to learn connection weights in an ANN. GAs are parallel search techniques that start with a set of random potential solutions and use special search operators (evaluation, selection, crossover, and mutation) to bias the search toward the promising solutions. At any given time, unlike any optimization approach, GA has several promising potential solutions (equal to population size) as opposed to one optimal solution. Each population member in a GA is a potential solution. A population member ($P_1$) used to learn the strength of connections for ANN shown in Figure 1 will consist of a set of all the connections. $P_1$ can be represented as:

$P_1 = \langle w_{14}, w_{15}, w_{16}, w_{24}, w_{25}, w_{26}, w_{31}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67} \rangle$

*Where* $(w_{14}, w_{15}, w_{16}, w_{24}, w_{25}, w_{26}, w_{31}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67}) \in \Re$

Any $w \in P_1$ is called a gene (connection weight) of a given population member $P_1$. A set of several population members is called a population $\Omega$. The cardinality of the set of population members $\Omega$ (number of population members) is called population size. The cardinality of a population (number of genes) member is called the defining length of the population member $\zeta$. The defining length for population member $P_1$, $\zeta = 12$. The defining length of all the population members in a given population is constant. GA starts with a random set of population. An evaluation operator is then applied to evaluate the fitness of each individual. In the case of learning connection weights for ANN for classification, the evaluation function is the number of correctly classified cases. A selection operator is then applied to select the population members with higher fitness (so that they can be assigned higher probability for survival). Under a selection operator, individual population members may be born, allowed to live or die. Several selection operators are reported in the literature; the operators are proportionate reproduction, ranking selection, tournament selection, and steady state selection (Goldberg and Deb 1991). Among the popular selection operators are ranking and tournament selection. Goldberg and Deb show that both ranking and tournament selection maintain strong population fitness growth potential under normal conditions. The tournament selection operator, however, requires lower computational overhead. The time complexity of ranking selection is *O (n log n)* whereas the time complexity of tournament selection is *O(n),* where n is number of population members in a population. In tournament selection, two random pair of individuals are selected and the member with the better fitness of the two is admitted to the pool of individuals for further genetic processing. The process is repeated in such a way that the population size remains constant and the best individual in the population always survives. For our research, we use the tournament selection operator.

After the selection operator is applied, the *new* population special operators, called crossover and mutation, are applied with a certain probability. For applying the crossover operator, the status of each population member is determined. Every population member is assigned a status as a survivor or non-survivor. The number of population members equal to survivor status is approximately equal to population size * (1 – probability of crossover). The number of non-surviving members are approximately equal to population size * probalility of crossover. The non-surviving members in a population are then replaced by applying crossover operators to randomly selected surviving members. Several crossover operators exists; we describe and use three different crossover operators in our research. The crossover operators used in our research are:

1. *One Point Crossover*. In one point crossover, two surviving parents and a crossover point are randomly selected. For each parent, the genes on the right hand side of the crossover point are exchanged to produce two children. Let $P_1$ and $P_2$ be two parents and the crossover point be denoted by "|". The two children $C_1$ and $C_2$ are produced as follows: (we use the bold font to simplify the understanding).

   $P_1 = \langle w_{14}, w_{15}, w_{16}, w_{24}, w_{25}, w_{26}, \mid w_{34}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67} \rangle$
   $P_2 = \langle \mathbf{w_{14}, w_{15}, w_{16}, w_{24}, w_{25}, w_{26}, \mid w_{34}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67}} \rangle$
   $C_1 = \langle w_{14}, w_{15}, w_{16}, w_{24}, w_{25}, w_{26}, \mathbf{w_{34}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67}} \rangle$
   $C_2 = \langle \mathbf{w_{14}, w_{15}, w_{16}, w_{24}, w_{25}, w_{26},} w_{34}, w_{35}, w_{36}, w_{47}, w_{57}, w_{67} \rangle$

2. *Uniform Crossover.* In uniform crossover, two surviving parents are randomly selected and flipping the genes in the two parents produces two children; probability of exchanging any given gene in a parent is 0.5. Thus, for every gene in a parent, a pseudo random number is generated. If the value of the pseudo random number is greater than 0.5, then the genes are flipped, else they are not flipped. If we have two random surviving parents, $P_1$ and $P_2$ (as shown in one-point crossover section), then a child, $C_1$, can be produced in the following manner:

   (a) Generate 12 (number equal to number of genes in the parent) pseudo random numbers between 0-1. Let these 12 random numbers be:
   $\langle$ 0.2, 0.3, 0.5, 0.6, 0.1, 0.7, 0.8, 0.8, 0.1, 0.9, 0.6, 0.1 $\rangle$

   (b) Flip the genes where a pseudo random number is greater than 0.5 as follows:
   $C_1 = \langle w_{14}, w_{15}, w_{16}, \mathbf{w_{24}}, w_{25}, \mathbf{w_{26}}, \mathbf{w_{34}}, \mathbf{w_{35}}, w_{36}, \mathbf{w_{47}}, \mathbf{w_{57}}, w_{67} \rangle$

3. *Arithmetic Crossover.* Arithmetic crossover consists of producing children in a way that every gene in a child is a convex combination of genes from its two parents. Given the two parents, $P_1$ and $P_2$ (as illustrated before), a child, $C_1$, can be produced as follows:
   $C_1 = \langle k_{14}, k_{15}, k_{16}, k_{24}, w_{25}, k_{26}, k_{34}, k_{35}, w_{36}, k_{47}, k_{57}, k_{67} \rangle$
   Where $k_{ij} = \lambda w_{ij} + (1-\lambda) \mathbf{w_{ij}}$, $\lambda \in [0,1]$ is a random number.

Arithmetic crossover ensures that every gene in the child is bounded by the respective genes from both parents. Unlike uniform and one point crossover, arithmetic crossover provides some local/hill climbing search (if the parents are on the opposite side of the hill) capability for a genetic algorithms. Arithmetic crossover is a popular crossover operator when GA is used for optimization (Pendharkar and Rodger 2000).

A mutation operator randomly picks a gene in a surviving population member (with the probability equal to the probability of mutation) and replaces it with a real random number.

In our experiments, we use all three crossover operators (one at a time) and investigate the performance of GA when different crossover operators are used. Thus, based on the crossover operator, we have three different types of GAs: genetic algorithm with arithmetic crossover called GA(A), genetic algorithm with uniform crossover operator GA (U), and genetic algorithm with one-point crossover operator GA (O). Our ANN architecture consists of four input nodes (3 inputs + 1 threshold), six hidden nodes (5 hidden + 1 threshold), and one output node. We use the same architecture as that of Bhattacharyya and Pendharkar (1998) so that we can benchmark the performance of our GA based training of ANN with the results of the back-propagation algorithm based ANN that Bhattacharyya and Pendharkar used. For our architecture, we have a population member defining length of $\zeta = 6$ ((3 inputs + 1 threshold) * 5 hidden + (5 hidden + 1 threshold) * 1 output).

## 3. A SHORT LITERATURE REVIEW AND HYPOTHESES

Several independent studies (Bhattacharyya and Pendharkar 1998; Joachimsthaler and Stam 1988; Koehler 1991; Koehler and Erenguc 1990) and found that data distribution characteristics determine the learning and predictive performance of different techniques for classification. Specifically, researchers found that variance heterogeneity and group distribution kurtosis affects the learning and predictive performance of the different techniques used for classification. We, therefore, propose the following two hypothesis (in the alternate form):

*H1:* *The group variance heterogeneity will have an impact on both the learning and predictive performance of the different techniques.*

*H2:* *The group distribution kurtosis will have an impact on both the learning and predictive performance of the different techniques.*

A back-propagation ANN uses a gradient descent algorithm to minimize RMS. Mathematically, this can be represented as:

$$Min\ E = \frac{1}{2}\sum_{n=0}^{n=N} \left\| t_n - o_n \right\|^2$$

The above minimization problem is an unconstrained minimization of a convex function. Gradient approaches are known to find the optimal solutions given the appropriate initial starting position (Pendharkar and Rodger 2000; Salomon 1998). GAs, on the other hand, are general-purpose optimization methods that use a survival of the fittest strategy to find heuristic solutions. For convex minimization based optimization problems, GAs are likely to under-perform the gradient based approaches (Pendharkar and Rodger 2000). This suggests that GA based ANN performance during the training phase will be lower than the back-propagation based ANN performance. This leads to our third hypothesis:

*H3: The back-propagation based ANN will have higher performance than the GA based ANN during the training phase.*

The arithmetic crossover, as described above, incorporates some hill climbing capabilities (when the two parents are on the opposite sides of the hill). Several researchers, because of the low disruption of schema (a specific pattern of genes in a population member), have used arithmetic crossover for the optimization problems (Pendharkar and Rodger 2000). In our case, GA using arithmetic crossover (because of its hill climbing nature arising from convex combination of two parents on the opposite sides of the hill) will have better performance than uniform and one-point crossover GAs during the learning phase. This leads to our fourth hypothesis:

*H4: The arithmetic-crossover GA based ANN will have higher performance than the uniform-crossover and one-point crossover GA based ANNs during the training phase.*

Pendharkar (1999) has observed that adding random noise to the connection weights during the back-propogation ANN decreases the performance of ANN during the training phases but improves the predictive performance of ANN. One of his arguments was that the gradient descent algorithms show high training performance and have a tendency to over-fit the training data sets (over-fitting is sometimes referred to as learning noise in the training data). Adding random noise to the connection weights during the training phase avoids the network over-fitting the training data and improves its performance on the test data. GA based ANN has a tendency not to over-fit the training data. The reason for over-fitting is that GA works with a population of potential solutions as opposed to one optimal solution approach used by most other techniques. At convergence, GA has a population of members that have similar fitness. The diversity of the fitness values in a population can be used to come up with a population member that has genes that are average of genes of entire population. This approach helps GA based ANN to learn the general patterns and avoid over-fitting the training data (Moriarty and Miikkulainen 1998). The generalized learning approach, without over learning the training data, is likely to perform better on test data set when compared to backpropagation ANN. This leads to our fifth hypothesis:

*H5: The GA based ANNs will outperform the backpropagation based ANN in the classification of unseen cases.*

Crossover consists of exchanging information from the two parents to produce children. Different crossover operators can be used to produce children. For one point crossover, there are $\zeta$-1 ways of crossover. For uniform crossover, there are $2^{\zeta}$ ways of crossover. Uniform crossover, because of its higher search space, has a potential to produce children that are less likely to contain the traits of any one parent. However, for promising two parents, one-point crossover, because of its restricted ways to crossover, is likely to produce promising children. Syswerda (1990), in his extensive experimental studies (using binary representation) found that uniform crossover under-performs one-point crossover for $\zeta < 29$ but, the performance improves when $\zeta > 30$. For $\zeta = 30$ both uniform and one-point crossover behave similarly. Rooiji, Jain and Johnson, in their empirical studies using non-binary representation, proposed following an upper bound heuristic of for the probability of the solution (schema) disruption for one-point crossover and uniform crossover:

$$P_d \le \frac{X}{L-1} \quad \textit{for 1-point crossover}$$

$$P_d \leq 1 - \left(\frac{1}{2}\right)^x \text{ for uniform crossover}$$

Where X is the number of inputs, $P_d$ is the probability of disruption, and L is defining length of the population member ($L = \zeta$). In our case, for X = 3 and L = 26 the upper bound for the probability of disruption for uniform crossover is higher than upper bound for the probability of disruption for one-point crossover. This leads to the following hypothesis:

*H6: Uniform crossover GA will under perform single point crossover GA for both learning and prediction.*

## 4. DATA SETS

For our research, we use data sets that have been previously used for comparing a number of techniques for classification. The data sets were first developed and used by Joachimsthaler and Stam (1988) to examine Fisher's linear discriminant function, the quadratic discriminant function, the logistic discriminant function, and linear programming approaches under varying group distribution characteristics. Koehler and Erenguc (1990), Banks and Abad (1991), and Abad and Banks (1993) used these data sets to establish experimental conditions to evaluate a number of linear programming approaches for the classification problem. In another study, Koehler (1991) used these data sets to determine the effectiveness of a linear genetic algorithm based discriminant analysis. Recently, Bhattacharyya and Pendharkar (1998) and Yanev and Balev (1999) used these data sets to compare a wide range of machine learning techniques and heuristic techniques for the classification problem. Pendharkar (1999) used these data sets to evaluate the impact of network architecture and input and weight noise on the learning and predictive performance of artificial neural networks.

The data sets consist of 1,200 data samples. Each data sample consists of three attributes and has 100 observations equally split between two groups. The data varies with respect to type of the distribution, determined through the kurtosis, and variance-covariance homogeneity (dispersion). Four kurtosis values of -1, 0, 1, and 3 correspond approximately to samples drawn from uniform, normal, logistic and Laplace population distributions. For dispersion variations across the data, if $I_i$ denotes the 3x3-dispersion matrix for Group i (i=1, 2), $I_1$ is always I (the identity matrix) while three different values are considered for the second group: $I_1$, 2 $I_1$, and 4 $I_1$. In order to minimize the effect of group overlap, the group means are set as follows: the group 1 mean is $\mu' = (0, 0, 0)$ throughout, and the group 2 mean was $\mu' = (.5, .5, .5)$ when $I_2 = I_1$, $\mu' = (.6, .6, .6)$ when $I_2 = 2I_1$ and $\mu' = (.8, .8, .8)$ when $I_2 = 4I_1$. There are thus 12 kurtosis-dispersion factor combinations leading to 12 data set groups. For each group, 100 random samples were taken, yielding a total of 1,200 data samples. A more detailed description of the data can be found in Joachimsthaler and Stam (1998).

## 5. EXPERIMENTAL RESULTS

We compare our three different types of GAs with back-propagation based ANN and genetic programming approaches of Bhattacharyya and Pendharkar (1998). We keep our experimental design same as that of Bhattacharyya and Pendharkar so that our results can be easily compared with their results on an equitable basis. Table 1 illustrates the training results of our experiments.

Tables 2 and 3 illustrate the results of three-way-ANOVA for training performance of different techniques. From Table 2, it can be seen that hypothesis 1 (variance heterogeneity) (F = 2739.30) and hypothesis 2 (distribution kurtosis ) (22.87) are supported (at 0.01 level of significance) for learning (training) performance of the techniques. Further, the interactions between variance heterogeneity and kurtosis and between technique and variance heterogeneity were significant (at level of significance = 0.01) as well. From Table 3, it can be seen that hypothesis 3 is supported (at 0.01 level of significance) with NN vs. GA (O) (F = 235.20), NN vs. GA (U) (F = 256.50) and NN vs. GA (A) (F = 206.42). No support to a very weak support (level of significance = 0.1) was found for hypothesis 4 (GA(A) vs. GA(O) and GA(A) vs. GA(U)) (F = 2.76).

Table 4 illustrates the results of testing the different techniques on the holdout samples. It can be seen that GA based ANNs outperform both back-propagation neural network and genetic programming. The difference in the performance between GA based ANN and back-propagation ANN decreases as the group variance heterogeneity increases.

Tables 5 and 6 illustrate the results of three-way-ANOVA on the holdout sample experiments. It can be seen from Table 5 that there is support (at level of significance 0.01) for hypotheses 1 (variance heterogeneity) (F = 3907.66) and 2 (distribution kurtosis) (F = 25.61) for holdout test data sets as well. Like the training results, the interaction effects of distribution kurtosis and variance heterogeneity were significant (at 0.01 level of significance) as well. However, unlike the training results, the interaction effect of distribution kurtosis and technique was significant at 0.05 level of significance. Further, Table 6 shows that hypothesis 5 is supported at 0.01 level of significance and the GA based ANN outperforms back-propagation based ANN (NN vs. GA (O) (F = 411.08), NN vs.GA (U) (F = 379.05), and NN vs. GA (A) (F = 366.80)). No support was found for hypothesis 6 (GA(U) vs. GA(O)) both in Tables 3 and Table 6.

Table 7 illustrates the results of CPU training run times (400 Mhz Pentium with 128 MB RAM) for our experiments. The CPU run times decreased as variance heterogeneity increased. This shows that population, with increase in group variance heterogeneity, converges quickly.

**Table 1. The Training Performance**

| Variance Heterogeneity | | Group Means | | Kurtosis | GA(A) | GA(O) | GA(U) | NN[a] | GP[a] |
|---|---|---|---|---|---|---|---|---|---|
| Group 1 | Group 2 | Group 1 | Group 2 | | Mean | Mean | Mean | Mean | Mean |
| 1 | 1 | 0 | 0.5 | -1 | 76.62 | 76.62 | 76.38 | 80.28 | 78.08 |
| 1 | 1 | 0 | 0.5 | 0 | 77.22 | 76.90 | 76.60 | 80.18 | 79.78 |
| 1 | 1 | 0 | 0.5 | 1 | 77.46 | 77.46 | 77.36 | 80.32 | 79.42 |
| 1 | 1 | 0 | 0.5 | 3 | 78.58 | 78.18 | 78.16 | 81.80 | 79.60 |
| 1 | 2 | 0 | 0.6 | -1 | 85.58 | 85.04 | 84.78 | 92.38 | 83.54 |
| 1 | 2 | 0 | 0.6 | 0 | 83.92 | 83.62 | 83.44 | 89.00 | 82.46 |
| 1 | 2 | 0 | 0.6 | 1 | 83.78 | 83.58 | 82.80 | 88.14 | 81.37 |
| 1 | 2 | 0 | 0.6 | 3 | 83.46 | 82.96 | 82.86 | 86.72 | 81.12 |
| 1 | 4 | 0 | 0.8 | -1 | 94.82 | 94.90 | 94.58 | 96.78 | 93.92 |
| 1 | 4 | 0 | 0.8 | 0 | 93.14 | 92.82 | 92.84 | 96.50 | 90.72 |
| 1 | 4 | 0 | 0.8 | 1 | 90.56 | 90.28 | 90.48 | 95.08 | 90.44 |
| 1 | 4 | 0 | 0.8 | 3 | 91.68 | 91.48 | 91.56 | 93.16 | 90.08 |

[a]Bhattacharyya and Pendharkar (1998)

**Table 2. The Correct Classification ANOVA Summary Table for Training Data**

| Source | Sum of Sq. | DF | Mean Sq. | F Ratio | P > F |
|---|---|---|---|---|---|
| *Main Effect* | | | | | |
| Distribution (D) | 1314.04 | 3 | 438.01 | 22.87 | 0.0001** |
| Variance (V) | 104928.34 | 2 | 52464.17 | 2739.30 | 0.0001** |
| Technique (T) | 7455.30 | 4 | 1863.82 | 97.32 | 0.0001** |
| *Two-Way-Interaction Effect* | | | | | |
| D × T | 161.15 | 12 | 13.43 | 0.70 | 0.7518 |
| D × V | 2312.50 | 6 | 385.42 | 20.12 | 0.0001** |
| T × V | 1846.56 | 8 | 230.82 | 12.05 | 0.0001** |
| *Three-Way-Interaction Effect* | | | | | |
| D × T × V | 524.99 | 24 | 21.87 | 1.14 | 0.2868 |

**Significant at 0.01 level of significance

**Table 3. The Overall Pairwise Comparisons on Training Data Sets**

| Contrast | DF | Sum of Sq. | F Value | P > F |
|---|---|---|---|---|
| NN vs. GP | 1 | 5100.56 | 266.31 | 0.0001** |
| NN vs. GA (O) | 1 | 4504.68 | 235.20 | 0.0001** |
| GP vs. GA (O) | 1 | 18.50 | 0.97 | 0.3258 |
| NN vs. GA (U) | 1 | 4912.65 | 256.50 | 0.0001** |
| GP vs. GA (U) | 1 | 1.76 | 0.09 | 0.7616 |
| GA (A) vs. GA (O) | 1 | 18.50 | 0.97 | 0.3258 |
| NN vs. GA (A) | 1 | 3945.81 | 206.02 | 0.0001** |
| GP vs. GA (A) | 1 | 74.00 | 3.86 | 0.0494* |
| GA (A) vs. GA (U) | 1 | 52.92 | 2.76 | 0.096 |
| GA (O) vs. GA (U) | 1 | 8.84 | 0.46 | 0.4969 |

*Significant at 0.05 level of significance
**Significant at 0.01 level of significance

**Table 4. The Holdout Sample Performance**

| Variance Heterogeneity | | Group Means | | Kurtosis | GA(A) | GA(O) | GA(U) | NN[a] | GP[a] |
|---|---|---|---|---|---|---|---|---|---|
| Group 1 | Group 2 | Group 1 | Group 2 | | Mean | Mean | Mean | Mean | Mean |
| 1 | 1 | 0 | 0.5 | -1 | 68.68 | 69.02 | 69.18 | 60.60 | 58.34 |
| 1 | 1 | 0 | 0.5 | 0 | 69.82 | 69.96 | 69.96 | 61.20 | 60.42 |
| 1 | 1 | 0 | 0.5 | 1 | 70.18 | 69.86 | 70.28 | 61.94 | 59.40 |
| 1 | 1 | 0 | 0.5 | 3 | 71.36 | 71.22 | 71.52 | 63.64 | 60.14 |
| 1 | 2 | 0 | 0.6 | -1 | 75.38 | 76.46 | 76.44 | 74.76 | 68.60 |
| 1 | 2 | 0 | 0.6 | 0 | 74.96 | 74.84 | 74.60 | 70.88 | 65.30 |
| 1 | 2 | 0 | 0.6 | 1 | 75.12 | 75.40 | 74.54 | 70.32 | 64.38 |
| 1 | 2 | 0 | 0.6 | 3 | 74.84 | 74.76 | 74.72 | 69.18 | 63.96 |
| 1 | 4 | 0 | 0.8 | -1 | 87.80 | 88.62 | 88.54 | 85.54 | 84.72 |
| 1 | 4 | 0 | 0.8 | 0 | 85.84 | 86.36 | 85.14 | 82.86 | 80.50 |
| 1 | 4 | 0 | 0.8 | 1 | 84.56 | 85.04 | 84.24 | 81.14 | 78.68 |
| 1 | 4 | 0 | 0.8 | 3 | 84.34 | 84.88 | 84.72 | 80.26 | 77.66 |

[a]Bhattacharyya and Pendharkar (1998).

**Table 5. The Correct Classification ANOVA Summary Table for Holdout Test Data**

| Source | Sum of Sq. | DF | Mean Sq. | F Ratio | P > F |
|---|---|---|---|---|---|
| *Main Effect* | | | | | |
| Distribution (D) | 1589.93 | 3 | 529.98 | 25.61 | 0.0001** |
| Variance (V) | 161724.79 | 2 | 80862.40 | 3907.66 | 0.0001** |
| Technique (T) | 37363.52 | 4 | 9340.88 | 451.40 | 0.0001** |
| *Two-Way-Interaction Effect* | | | | | |
| D × T | 528.24 | 12 | 44.02 | 2.13 | 0.0128* |
| D × V | 3715.26 | 6 | 619.21 | 29.92 | 0.0001** |
| T × V | 3711.33 | 8 | 463.92 | 22.42 | 0.0001** |
| *Three-Way-Interaction Effect* | | | | | |
| D × T× V | 429.87 | 24 | 17.91 | 0.87 | 0.6518 |

*Significant at 0.05 level of significance
**Significant at 0.01 level of significance

**Table 6. The Overall Pairwise Comparisons on Holdout Test Data Sets**

| Contrast | DF | Sum of Sq. | F Value | P > F |
|---|---|---|---|---|
| NN vs. GP | 1 | 3474.80 | 167.92 | 0.0001** |
| NN vs. GA (O) | 1 | 8506.69 | 411.08 | 0.0001** |
| GP vs. GA (O) | 1 | 22855.14 | 1104.47 | 0.0001** |
| NN vs. GA (U) | 1 | 7843.85 | 379.05 | 0.0001** |
| GP vs. GA (U) | 1 | 21760.08 | 1051.55 | 0.0001** |
| GA (A) vs. GA (O) | 1 | 26.11 | 1.26 | 0.2614 |
| NN vs. GA (A) | 1 | 7590.27 | 366.80 | 0.0001** |
| GP vs. GA (A) | 1 | 21336.33 | 1031.07 | 0.0001** |
| GA (A) vs. GA (U) | 1 | 2.08 | 0.10 | 0.7510 |
| GA (O) vs. GA (U) | 1 | 13.44 | 0.65 | 0.4203 |

**Significant at 0.01 level of significance

**Table 7. The CPU Run Time in Seconds**

| Variance Heterogeneity | | Group Means | | Kurtosis | GA (A) | GA (O) | GA (U) |
|---|---|---|---|---|---|---|---|
| Group 1 | Group 2 | Group 1 | Group 2 | | Mean (SD) | Mean (SD) | Mean (SD) |
| 1 | 1 | 0 | 0.5 | -1 | 67.1 (0.49) | 65.3(0.63) | 69.4(5.11) |
| 1 | 1 | 0 | 0.5 | 0 | 68.7(4.43) | 66.9(0.49) | 66.9(0.54) |
| 1 | 1 | 0 | 0.5 | 1 | 68.5(5.27) | 66.8(0.56) | 68.4(4.79) |
| 1 | 1 | 0 | 0.5 | 3 | 67.1(0.56) | 68.4(4.97) | 67.1(0.62) |
| 1 | 2 | 0 | 0.6 | -1 | 64.0(4.81) | 62.6(1.18) | 62.4(0.48) |
| 1 | 2 | 0 | 0.6 | 0 | 64.2(4.76) | 62.5(0.69) | 64.1(4.99) |
| 1 | 2 | 0 | 0.6 | 1 | 64.1(2.44) | 62.7(0.78) | 64.12(4.91) |
| 1 | 2 | 0 | 0.6 | 3 | 63.1(0.87) | 64.3(4.88) | 63.1(0.88) |
| 1 | 4 | 0 | 0.8 | -1 | 56.4(0.59) | 57.6(4.63) | 56.2(0.55) |
| 1 | 4 | 0 | 0.8 | 0 | 56.8(0.74) | 57.9(4.26) | 56.7(0.74) |
| 1 | 4 | 0 | 0.8 | 1 | 57.4(1.95) | 58.6(5.43) | 57.3(1.92) |
| 1 | 4 | 0 | 0.8 | 3 | 57.5(0.90) | 57.4(0.92) | 58.9(4.88) |

## 6. DISCUSSION AND CONCLUSIONS

Our experiments show that GA based ANN training shows resistance toward over-fitting in a binary classification problem. We think that this is a strong facet and researchers and practitioners should use GA based ANN when a higher predictive performance is desired. Our study was one of the first to compare the performance of different crossover operators related to design of GA based ANN. Although no significant difference was found between the different crossover operators, we believe that for larger networks crossover may play a vital role during learning (since GA(A) vs. GA(U) were significantly different at level of significance = 0.1 during learning phase).

In our experiments, we kept the ANN architecture constant. Several studies have shown that the ANN architecture plays an important role during the training and predictive performance. In cases where higher learning performance is desired, hybrid approaches might have potential. For example, the best fitness population member from the GA based ANN can be used as an initial set of weights for back-propagation algorithm and higher learning performance may be obtained. The hybrid approaches may have a merit when compared to random initialization of weights for an ANN. Future research may focus on the impact of network architecture on the training and predictive performance of GA trained ANN. Hybrid approaches deserve merit for future investigation as well.

## 7. ACKNOWLEDGMENTS

The research was supported, in part, by the Capital College research council grant of The Pennsylvania State University. We thank the associate editor, the track co-chairs, and the anonymous reviewers for their valuable comments and insights into this research.

## 8. REFERENCES

Abad, P. L., and Banks, W. J. "New LP Based Heuristics for the Classification Problem," *European Journal of Operations Research* (67), 1993, pp. 88-100.

Banks, W. J., and Abad, P. L "An Efficient Optimal Solution for the Classification Problem," *Decision Sciences* (22:5), 1991, pp. 1008-1023.

Bhattacharyya, S., and Pendharkar P. C. "Inductive Evolutionary and Neural Techniques for Discrimination," *Decision Sciences* (29:4), 1998, pp. 871-899.

Curry, B., and Morgan, P. "Neural Networks: A Need for Caution," *Omega: The International Journal of Management Science* (25:1), February 1997, pp. 123-133.

Goldberg, D. *Genetic Algorithms in Search, Optimization, and Machine Learning*, Reading, MA: Addison Wesley Longman, Inc., 1989.

Goldberg, D. E., and Deb, K. "A Comparative Analysis of Selection Schemes Used in Genetic Algorithms," in *Foundations of Genetic Algorithms*, G. Rawlins (ed.), San Mateo, CA: Morgan Kaufmann, 1991, pp. 69-93.

Hung, M. S., and Denton, J. A. "Training Neural Networks with GRG2 Nonlinear Optimizer," *European Journal of Operational Research* (69), 1993, pp. 83-91.

Janikow, C. Z., and Michalewicz, Z. "An Experimental Comparison of Binary and Floating Point Representations in Genetic Algorithms," in *Fourth International Conference on Genetic Algorithms*, 1991, pp. 31-36.

Joachimsthaler, E. A., and Stam, A. "Four Approaches to the Classification Problem in Discriminant Analysis: An Experimental Study," *Decision Sciences* (19), 1988, pp. 322-333.

Koehler, G. J. "Linear Discriminant Functions Determined by Genetic Search," *ORSA Journal on Computing* (3:4), Fall 1991, pp. 345-357.

Koehler, G. J., and Erenguc, S. S. "Minimizing Misclassifications in Linear Discriminant Analysis," *Decision Sciences* (21), 1990, pp. 63-85.

Miller, G. F.; Todd, P. M.; and Hedge, S. U. "Designing Neural Networks using Genetic Algorithms," in *Proceedings of Second International Conference of Genetic Algorithms*, 1989, pp. 379-383.

Moriarty, D. E., and Miikkulainen, R. "Forming Neural Networks Through Efficient and Adaptive Coevolution," *Evolutionary Computation* (5:4), 1998, pp. 373-399.

Pendharkar, P. C. "Neural Approaches in Noisy and Kurtotic Environments," under review, 1999 (available from the first author).

Pendharkar, P. C., and Rodger, J. A. "A Non-Linear Programming and Genetic Search Application for Production Scheduling in Coal Mines," *Annals of Operations Research*, forthcoming, 2000.

Rooij, A. J. F.; Jain, L. C.; and Johnson, R. P. *Neural Network Training Using Genetic Algorithms*" Series in Machine Perception and Artificial Intelligence, Volume 26, Singapore: World Scientific Publishers, 1998.

Salomon, R. "Evolutionary Algorithms and Gradient Search: Similarities and Differences," *IEEE Transactions on Evolutionary Computation* (2:2), 1998, pp. 45-55.

Sexton, R. S.; Alidaee, B.; Dorsey, R. E.; and Johnson, J. D. "Global Optimization for Artificial Neural Networks: A Tabu Search Application," *European Journal of Operational Research* (106), 1998, pp. 570-584.

Sexton, R. S.; Dorsey, R. E.; and Johnson, J. D. "Optimization of Neural Networks: A Comparative Analysis of the Genetic Algorithm and Simulated Annealing," *European Journal of Operational Research* (114), 1999, pp. 589-601.

Syswerda, G. "Uniform Crossover in Genetic Algorithms," in *Proceedings of Third International Conference on Genetic Algorithms*, 1990, pp. 2-9.

Whitley, D., and Hanson, T. "Optimizing Neural Networks Using Faster, More Accurate Genetic Search," *Proceedings of Second International Conference on Genetic Algorithms*, 1989, pp. 391-396.

Wilson, S. W. "GA: Easy Does Not Imply Steepest-Ascent Optimizable," in *Proceedings of Fourth International Conference on Genetic Algorithms*, 1991, pp. 85-89.

Yanev, N., and Balev, S. "A Combinatorial Approach to the Classification Problem," *European Journal of Operational Research* (115), 1999, pp. 339-350.